Interactive Design of 3-D Computer-Animated Legged Animal Motion

Michael Girard*
Ohio State University
Computer & Information Science Dept.
Computer Graphics Research Group
Columbus, OH 43201

November 19, 1986

Abstract

We present a visually interactive approach to the design of 3-D computer-animated legged animal motion in the context of the PODA computer animation system. The design process entails the interactive specification of parameters which drive a computational model for animal movement. The animator incrementally modifies a framework for establishing desired limb and body motion as well as the constraints imposed by physical dynamics (newtonian mechanical properties) and temporal restrictions. PODA uses the desired motion and constraints specified by the animator to produce motion through an idealized model of the animal's adaptive dynamic control strategies.

*This research was supported by a National Science Foundation grant DCR-8304185.

1 Introduction

In contrast to inanimate objects, animals plan their motion in accordance with their postural goals. Computational models of legged animal motion, in the context of computer animation, must produce the visual appearance of the coordinated motion which results from the planning strategies animals use to achieve their goals. The interactive design of kinematic motion and its integration with limb and body dynamic control strategies form the basis of our discussion. We begin with a description of the interactive specification of limb motion before embarking on the problem of designing animation of animal locomotion and finally, non-periodic animal dance.

2 Interactive Design of Limb Motion

In a broad sense, a limb's motion may be characterized by its path and associated time

derivatives in either joint space or cartesian space. The joint-angle interpolation strategy adopted by "keyframe" computer animation systems operates solely in joint-space — the path, speed, and acceleration of the limb's end-effector (hand or foot) may not be precisely controlled. This is a serious deficiency given the importance of coordinated end-effector motion for the manual manipulation of objects and placement of feet. Furthermore, as we shall discuss in section 2.6, empirical evidence suggests that end-effector based trajectory planning may be operative in the coordination of coordinated unrestrained animal limb movements.

The usual joint-angle specification of a key posture for a limb is augmented by the position/orientation of the end-effector (hand or foot). Limbs (and the spine) may be interactively manipulated into key positions using both forward and inverse kinematics.

Because the animal limbs we wish to model typically have limb geometries which have more than six degrees-of-freedom, we must adopt inverse-kinematic techniques suitable for the class of *redundant* limbs. We have used *resolved motion rate* or *pseudo-inverse jacobian* control for this purpose. This technique linearizes the kinematic equations of motion about a point and solves for the position of the end-effector in terms of its velocity or "rate" [1–4]. A recent publication [5] presents a closed-form analytical solution which yields the end-effector position directly, but results in a set of non-linear equations which must be solved numerically.

Inverse-kinematic control of redundant limbs takes the following simplified form:

$$\theta = F(X, \Lambda)$$

where X is the desired cartesian-space position of the end-effector
Λ is the desired joint-space position
θ is the actual joint-space position
and F is the inverse-kinematic function

The solution that places the end-effector at the desired cartesian position X is found which, as a secondary goal, minimizes its deviation from the desired joint-space position Λ.

2.1 Interactive Limb Positioning

The interactive positioning of limbs in PODA may be accomplished by either movement of the end-effector, rotation of the joints, or rotation of the joints with the end-effector stationary (when possible). Movement of the end-effector results when the user alters the cartesian coordinates or orientation of the desired position X. In this mode, the program solves the inverse-kinematics to bring the end-effector to the new desired position. This mode is essential for placing feet and hands at specific places in the environment, e.g., placing the feet on the ground, putting the hands on the hips, or cases involving reaching behavior.

There are times when it is more convenient to move a limb by changing its joint angles rather than positioning its hand or foot. This is particularly true for forming postures which are meant to occur during joint-interpolated movements, such as the swinging

of arms or the kicking of legs. The user may select and move a joint forward and backward. In such cases, the program employs forward kinematics to update the limb and its new end-effector position.

The third mode allows the user to fine-tune the posture of a limb once its end-effector has been constrained to be anchored to some desired position. This is achieved by adjusting the desired joint angles in Lambda and then solving the inverse-kinematics problem. Only the joints will realign themselves as long as the desired position, X, remains fixed.

The user may freely alternate between these three modes in order to converge upon some desired limb configuration.

2.2 Composition of limb posture sequences

Our approach defines a limb trajectory in terms of:

1. the path which the limb negotiates,
2. a function that controls the speed of the limb along its path.

Before embarking on the problem of controlling the speed of movement, we will first describe the interactive specification of the limb's path. The path is defined by a sequence of limb postures, called the *posture sequence*, which the animator desires the limb to pass through. A *posture* is composed of a limb's position in both cartesian-space (the posture's end-effector position) and joint-space (the posture's joint-angles). The actual path taken through these postures may be a function of spline-interpolation of the joint angles or a 3-D interpolating-spline which passes through the end-effector coordinates of each posture in the sequence.

In PODA, each limb has two associated tables which act as posture and path "memories." A *posture table* stores postures which the animator would like to retrieve at a later time. This is necessary for cyclic motions wherein we wish the limb to return to previously assumed postures. The animator may append, overwrite, retrieve and delete postures from the limb's posture table by using a menu-driven interface.

The animator assembles postures into a *working posture sequence*. An interactive menu provides the animator with a means of appending, inserting, replacing, and deleting postures from the working sequence. Once a satisfactory working sequence has been created, it may be stored in the limb's *posture sequence table*. Sequences of postures which represent different limb paths are remembered and subsequently recalled from this table. As with the posture table, the animator may append, replace, retrieve and delete posture sequences from this table.

2.3 Speed control by Arc-length Reparameterization

A posture sequences roughly describes the path the limb is to follow. The *speed* the limb moves along that path must also be specified. There have been a number of attempts in the computer animation field to deal with this problem. In parametric key-frame animation, interpolating splines are employed to specify position as a function of the spline's parameter: $spline(u) = p$. Typically time

is assumed to be proportional to the parameter u of the interpolating spline. That is, $spline(u(t)) = p$ where $u(t) = Kt$. Kochanek has shown that one may vary the tangential speed along Catmull-Rom cubic splines about a control point by changing the magnitude of the tangent vectors at that point, a parameter she calls "tension" [8]. However, the shape of the path is also altered in the process.

Steketee and Badler introduced a "double-interpolant" method which employs a B-spline curve to represent $u(t)$, thereby allowing variation from the usually assumed linear relationship. Although their technique provides for the variation of *parametric* speed, $\|du(t)/dt\|$, this may be dramatically different from the actual speed in the geometric sense, $\|d(spline(u))/dt\|$.

A central problem in using parametric splines for the specification of motion is that equal lengths in the parameter u will not, in general, map into equal increments along the curve in euclidean space. Unevenly spaced control points will result in variations of speed along the curve. What is needed for precise control of speed is a *reparametrization by arc-length* of the interpolating spline. If the function $h(u)$ gives the arc-length of the curve, the parameter s, the inverse of h, will yield u values which map equal increments of s into equal increments along the curve [see fig. 1]. The reparametrized curve $reparSpline(s) = spline(h^{-1}(s))$ has the property that $\|d(reparSpline)/ds\| = 1$ [7].

Although a reparametrization by arc-length for any regular curve theoretically exists, in practice we may not be able to find an analytical solution. Unfortunately, this is the case

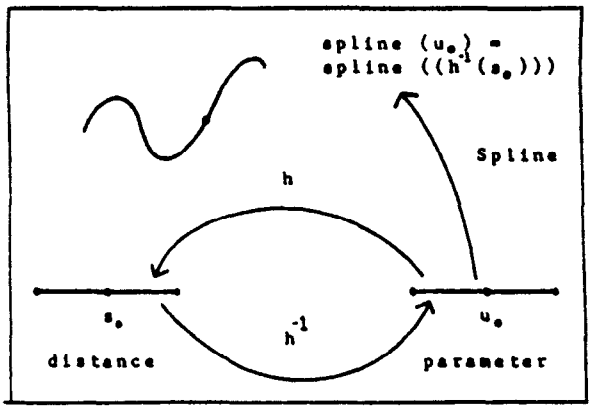

Figure 1:

for spline curves. The norm of the derivative of the spline (the tangent vector) must be integrated and then inverted.

Since this is not practically solvable, we must use numerical techniques to find h [6]. We use a gaussian quadrature program optimized for speed by Brian Guenter for this purpose. Of course, this is not sufficient since we still need h^{-1}. We approximate h^{-1} by first creating a look-up table with entries for arc-length h for small increments in the spline parameter u. We may then get the inverse of h by applying linear interpolation to the table.

With the use of the h^{-1} table, we may now precisely control both the distance and speed of an object (in our case, a limb) which travels along its path.

2.4 Interactive Design of Speed

Although we may control speed independently of path, in the context of limb trajectory motion, one typically wants to design the speed function in relation to postures

which define the path. If speed were designed solely over time, we could not easily control how changes in acceleration occurred about specific postures. The animator requires a means of designing speed in relation to specific distances associated with postures alongs the limb's path.

Our solution is to provide the animator with a *distance(time) graph* control mechanism, a graph which represents distance as a function of time. In such a graph, the vertical axis represents distance. Since the postures in the posture sequence of the limb occur at known distances along the path, we may represent these distances by placing horizontal dotted-lines along the vertical axis [see fig. 2]. Since the horizontal axis is time, the derivative with respect to time $d(dist)/dt$ — the tangent vector along the graph — represents speed. Seen in this way, the animator may adjust the speed about specific postures by altering the slope of the graph as it crosses the horizontal lines corresponding to distances at which the postures fall.

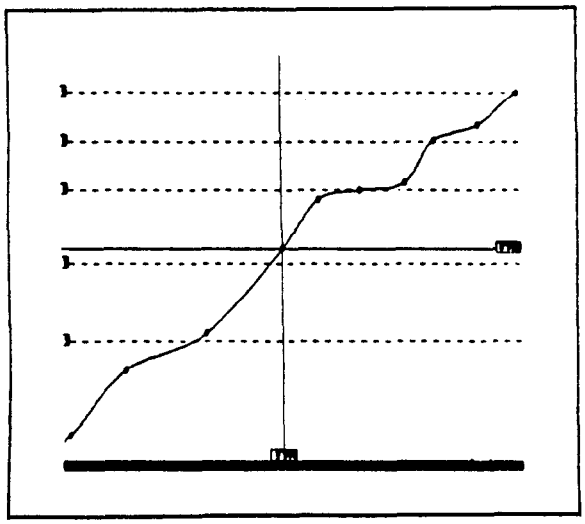

Figure 2:

The points of inflection on the distance(time) graph are analogous to extrema on a speed(time) graph. We note that if we scale the distance(time) graph in the time dimension, the points of inflection will still occur at the same distances. The net effect is that the speed will change in proportion to the degree of scaling in time without changing the relative acceleration in relation to distance. Therefore, the distance(time) graph may be normalized to an arbitrary range. The character of motion is preserved when we squeeze or stretch the graph to accommodate any desired duration.

2.5 Graphs from 2-D Splines

In order for the animator to have fine control, our distance(time) graph/function must be amenable to modification. In effect, we must be able to design the shape of this graph. The "double interpolant" method introduced by Steketee and Badler employed a 2-D spline, or "kinetic spline" as a means of specifying a $u(time)$ graph. We apply their method to the shaping the distance(time) graph. Note that their procedures for creating smooth transitions in the "parametric speed" between two motions may be applied equally well in our context for transitions in actual "geometric speed."

Transforming a 2-D spline into a graph requires that we interpret a point (x, y) on the 2-D curve curve as (time, distance(time)). Once again we are faced with the problem that equal increments in spline(u) will

not map into equal increments in x. That is, for time t_0, we must find u such that $spline(u)_x = t_0$. Then $spline(u)_y$ is the distance at t_0. Numerical methods (such as Newton-Raphson) may be applied to solve for u with $spline(u)_x - t_0 = 0$.

2.6 Modeling Natural Limb Motion

So far, we have described a model for designing motion having any characteristics the animator desires. The trajectories need not conform to any laws of behavior which govern motion in the real world. This is fine if we are not concerned with producing limb motion which appears to look natural. But if we are, we must look for higher-level models of limb movement which embody the dynamics and control strategies employed by real animals. The design of limb motion should be supported by a parametric model which captures the essential patterns of natural limb behavior.

The specification and control of limb trajectories is an active area of research in the robotics field. The problem of *trajectory planning* is to synthesize a nominal path and speed function which moves a limb (or industrial manipulator) between desired postures. In real physical situations, one is interested in solving for the trajectory of the limb in terms of some optimization criteria (such as minimum-time, minimum-jerk or minimum energy) in the presence of dynamic and kinematic constraints (such as maximum achievable joint torques, kinematic joint limits and environmental obstacles) [20].

Empirical studies have been conducted to determine how humans and other animals solve the trajectory planning problem [11–16]. Especially noteworthy are the analyses of unrestrained arm movements in humans [11, 14]. It was discovered that the normalized speed(time) graph of the hand's motion between two stationary points in space was invariant with respect to the points chosen, the load carried, the speed of motion and the distance traveled. Furthermore, the shape of this graph matched that predicted by an *optimization for minimum-jerk about the hand.* In general, the empirical studies suggest the existence of a hierarchy of control which separates the planning of trajectories from the actuation of forces in the limb which must take into account the actual dynamics of motion [see fig. 3].

We are currently engaged in implementing trajectory planning based on the optimization criteria found in the empirical studies. The goal is to automatically generate the distance(time) graph of a motion and path from a posture sequence. In theory, it should be possible to determine the extent to which the path of motion is resolved in joint-space or cartesian-space and the shape of the distance(time) graph as a weighting of various dynamic optimization criteria [13].

In the computer animation field, there have been attempts to animate articulated limbs and animals with the use of *forward dynamics* [17–19]. In this approach, one simulates the physical properties of the limb and solves its motion from the torques applied at each of the joints. This reduces the problem to finding the torque(time) function, at each joint, which produces some desired limb motion.

But if these torque(time) functions are gen-

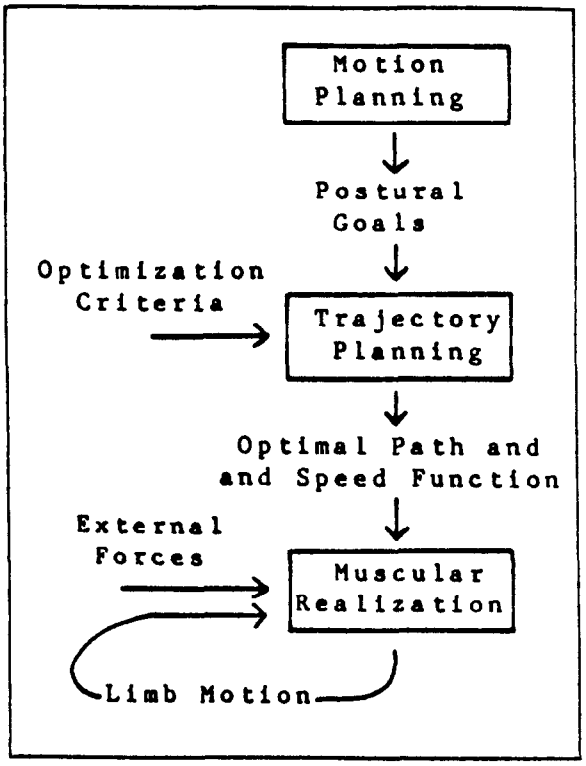

Figure 3:

erated to adhere to purely kinematic trajectory motion plans, as suggested above, the modeling of dynamics may be both unnecessary and insufficient for the animation of naturalistic coordinated limb motion. However, if the optimization criteria involves functions for forces, such as the minimization of energy (which may be operative during the swinging of limbs), we must then incorporate dynamics into the trajectory planning computation.

We argue that computational models of animal motion must embody control strategies which optimize performance by planning nominal trajectories in advance of the actual movement [10]. In general, the control of coordinated actions by real animals frequently involves a planning stage wherein the animal *anticipates* its future postural positions.

3 Integration of Limb and Body Motion

The notion of planning movement to satisfy goals and constraints in terms of known future states in used excessively by PODA in the execution of coordinating body and limb motion. In the following, we will describe our work toward building models of animal locomotion with periodic gaits and the performance of non-periodic dance.

3.1 Leg Motion Adaptations to Body Motion

Variations in leg motion occurs between gaits, between animals, and between legs. For example, consider human locomotion. During walking the legs swing and touch down on

the heel; running is often distinguished by a "kick" of the legs which brings the runner's foot high above the knee as his foot leaves the ground. The limb trajectory tables, described above, provide the animator with a means of defining a host of trajectories which may be used to distinguish the character of many different types of leg stepping motions. PODA provides an interface which allows the animator to associate a trajectory from a leg's trajectory table with a given gait.

Since limb trajectories are designed in the coordinate space of the legs' hip, they must be adapted during body movement to accommodate variations in footholds and body speed. Footholds are calculated to bring the leg to a stable reference position halfway through the leg's support period (for details regarding this computation, see [21]).

Another problem is that we must prevent discontinuities in foot velocity during the foot's transitions between being on and off the ground. Jumps in speed are quickly seen to appear unnaturally jerky. Real physical legs must "follow through" as they lift from the ground due to their momentum backward during the support phase. PODA is able to simulate the appearance of a gradual change in momentum by measuring the leg's joint-velocities as the foot leaves the ground. A joint-space trajectory may be described by maintaining the measured lift-off joint speed. We sinusoidally interpolate from this constant joint-velocity trajectory to our leg trajectory selected from the table.

When a foot is placed on the ground, real animals form leg trajectories which minimize dramatic changes in horizontal velocity with respect to the ground. This reduces the muscular stress of absorbing sudden accelerations which might occur at the point of contact. PODA alters the selected leg trajectory toward the end of its transfer phase by sinusoidally interpolating its horizontal foot position toward its world-space location in the preceding frame. This, in effect, moves the foot toward the direction and speed of the ground moving under the animal's body. At the point of contact, the foot will accelerate smoothly to match the velocity of the ground.

3.2 Periodic Gait Specification

For purposes of animating animal locomotion, the user specifies the desired body path in the horizontal plane (with a spline) and a sequence of gaits to be performed. For our present discussion, we must review the gait parameters of PODA's locomotion model [21].

A *gait pattern* describes the sequence of lifting and placing of the feet. The pattern repeats itself as the animal moves: each repetition of the sequence is called the *gait cycle*.

The time (or number of frames) taken to complete a single gait cycle is the period, P of the cycle.

The relative phase of leg i, R_i, describes the fraction of the gait cycle period which transpires before leg i is lifted [see figure 4].

During each gait cycle period any given leg will spend a percentage of that time on the ground–this fraction is called the *duty factor* of leg i.

We will call the time a leg spends on the ground its *support duration*. The time spent in the air is the leg's *transfer duration*.

The *stroke* is defined as the distance trav-

eled by the body during a leg's support duration.

3.3 Leg Coordination

The following equalities must hold by definition for any leg:

$$supportDuration = stroke/bodySpeed$$
$$P = supportDuration/dutyFactor$$
$$transferDuration = P - supportDuration$$

The gait cycle period for each leg is best imagined as a duration subdivided into support and transfer duration.

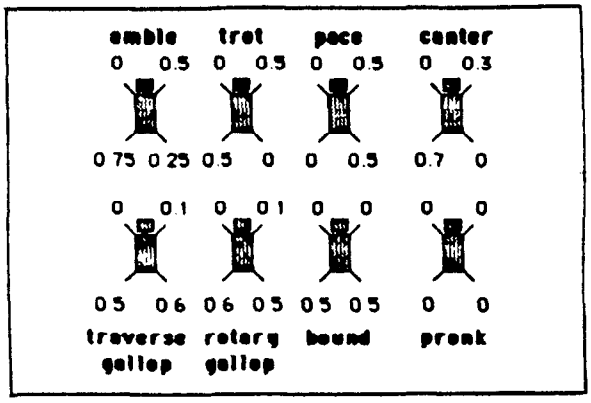

Figure 4: Relative phase relationships for several quadruped gaits

The *leg state* at time t may be determined as

$$legState = legState_0 + t \bmod P$$

where $legState_0 = R_i \times P$

If the leg state is less than the support duration then the leg is in its support phase, otherwise the leg is in its transfer phase. Moreover, the time of foot placement occurs when the leg state equals zero and the foot liftoff occurs when the leg state is equal to the support duration [see fig. 5].

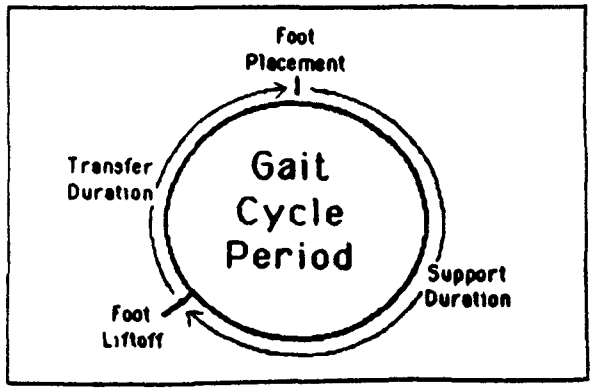

Figure 5:

We see that each gait may be uniquely defined by a body speed, leg dutyFactor, stroke, and the relative phase relationships between the legs. The animator interactively sets the values of these parameters for each gait in a sequence of gaits defined using a menu-driven interface.

3.4 Gait Shifting

Legged animals frequently change their gaits during locomotion. These changes are most often a function of body speed. The well-known sequence of gaits executed by accelerating horses — amble, trot, canter, gallop — is common to many four legged animals. But rather than limit ourselves to a fixed set of gait shifts, we sought to develop a model which will animate transitions between arbitrary sequences of gaits at arbitrary speeds.

To do so, it was necessary to implement a gait shifting algorithm which could adjust the coordination of legs between any two gaits. The problem is to shift the phase of the legs,

either by reducing or increasing the time the legs spend during their transfer phase. Applying phase shifts during the support phase was ruled out due to its possibly unnatural or potentially impossible requirements on the length of the stride (recalling that the stroke equals the product of the support duration and the body speed).

Since there is no unique solution for a set of phase shifts between two gaits, an optimization criteria had to be applied. We chose to minimize the square of the magnitudes of the phase shifts under the constraint that the resulting adjusted leg transfer durations should not be shorter than a given constant. That is, we wish to minimize

$$relativePhaseDist = \sum_{i=1}^{n} phaseShift^2$$

such that

$$transferDuration_i + phaseShift_i \geq k$$

where

$1 \leq i \leq n$
n = number of legs
k = minimum leg transfer time

Minimization of phase-distance has the effect of solving for the smallest degree of shifting which is evenly distributed among the legs. The minimum leg transfer constraint prevents solutions which will require unnaturally quick leg motions.

First, let us consider the problem for cases in which the gait cycle period is the same for both gaits. For our computer-animated gaits, gait cycle periods are adjusted to fall on integer boundaries so that liftoff and touchdown times occur exactly at specific frames. This reduces the number of possible phase shifts to $2 * numberoflegs * period$ if we consider all possible differences between the legStates of two gaits. Given the small number of possible candidates, PODA performs a simple exhaustive search.

When the current and next gaits have different periods, a two stage transition is employed. In the first stage, a phase shift is performed to equate the gait-cycle periods in terms of the longest period.

If the current period is shorter,

1. the current gait is shifted to a longer period by shifting all legs by the difference in gait cycle periods.
2. the algorithm for equal gait cycle periods is applied.

If the current period is longer, steps 1 and 2 are reversed.

4 Body Motion Trajectories as a function of support profiles

A critical factor in the determination of an animal's body motion is a description of its history of contact (or lack of contact) with the ground. PODA computes a record of the support and non-support intervals of the animal as a first step in simulating its body motion. The sequence of these intervals forms the *support profile* of the animal.

Currently, animals in PODA must have an even number of legs. Quadrupeds (four legged animals) are supplied with two support

profiles — one for each pair of front and hind legs. PODA may then simulate the pitching motions of quadrupeds by isolating the vertical motion of the shoulder and pelvis [see fig. 6]. PODA records only one support profile for bipeds (two legged animals) since they have only one pelvis. Animals with more than four legs are typically rigid (insects), so one support profile is used for them as well.

The building of the support profile is the first stage in the computations for synthesizing animal locomotion. PODA incrementally derives the animal's body trajectory by a four pass process of accumulating constraints on its three-dimensional degrees-of-freedom.

1. the support profile is constructed as a function of the gaits and gait transitions,

2. the vertical and horizontal motion of the body may be computed as a function of the support profiles,

3. the angular motion necessary for keeping the body pointed toward its horizontal direction of movement and the angular banking required to accommodate the curvature of the horizontal path may be calculated once its translational positions at each frame are known,

4. the kinematic motion of the limbs and the calculation of footholds may be computed as a function of the known body positions at each frame. [see fig. 7].

The record of future positions derived from each pass allows PODA to mimic the body trajectory planning that animals exhibit in the performance of coordinated body move-

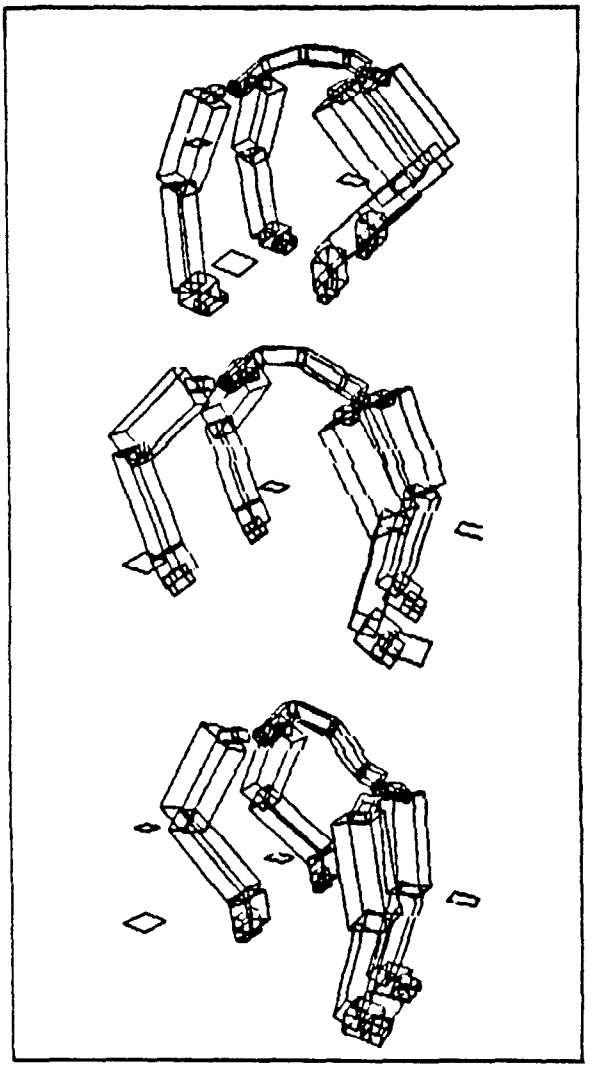

Figure 6: Pitching motion of quadruped

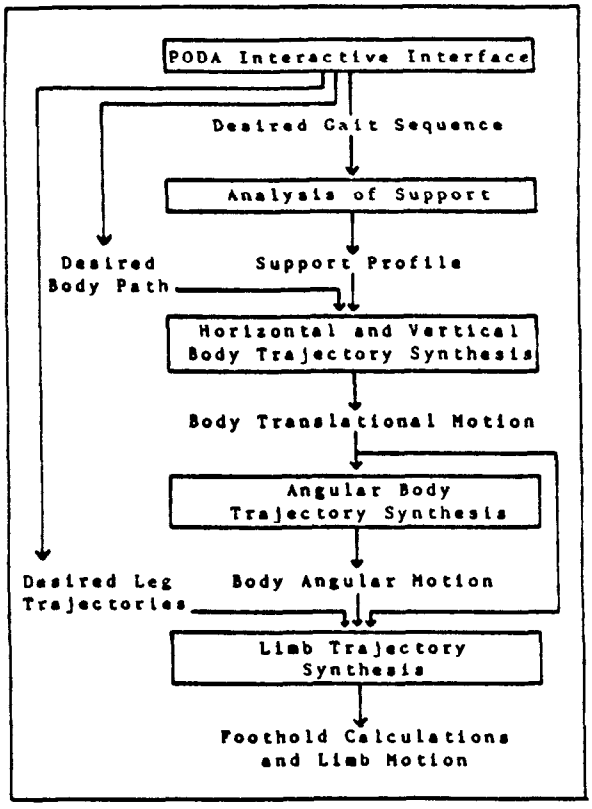

Figure 7:

ment. In the following, we will discuss the approach taken for each pass of this process.

4.1 Vertical Body Motion

During the non-support intervals of the support profile, acceleration on the animal is restricted to the effects of gravity. Given the downward gravitational acceleration the body must leave the ground at an upward velocity required to keep it in the air for the duration of each non-support interval. PODA solves Newton's equations of motion for the upward velocity and the resulting ballistic motion of the body during its non-support periods. The final downward vertical speed at the end of each non-support interval is also easily calculated.

The problem of synthesizing the trajectory of the body during its support intervals remains. We must solve for the trajectory under the constraints that:

1. the initial and final vertical speeds must be equal to the values computed for the non-support intervals,
2. the vertical speed must be continuous,
3. the liftoff height must be less than or equal to the length of the leg.

The second constraint prevents the appearance of unnaturally stiff movement in which strong accelerations would have to occur in between frames (such as a bouncing pool ball). The third constraint restrains pelvic movement to remain inside the kinematic boundaries of the leg.

The solution employed by PODA is to describe the velocity curve in terms of a family

of functions of the form At^e where $0 < t \leq (1/2)(supportInterval)$ and $e \geq 1$ [see fig. 8] In this context, each of the constraints im-

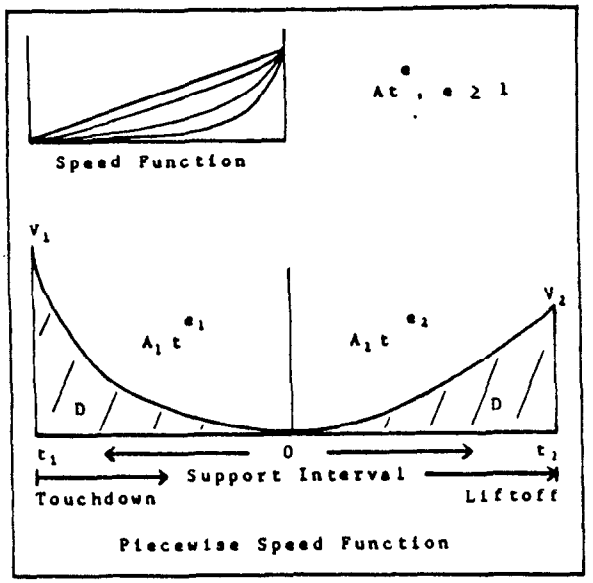

Figure 8:

plies, respectively:

1. $A_1 t_1^{e_1} = v_1$
   $A_2 t_2^{e_2} = v_2$
   where $t_1 = t_2 = (1/2)(supportInterval)$
   v_1 = mag. of initial downward speed
   v_2 = mag. of final upward speed

2. *Given $e_i \geq 1$, we may join the speed curves piecewise-continuously at speed = 0, with the first function reversed.* [See fig. 8]

3. $\int_0^{t_i} A_i t^{e_i} dt \leq D$

where D is the distance between maximum and minimum vertical heights.

If we solve these equations with equation 3 set to the equality, we get:

$$e_i = \left[\frac{(V_i)(t_i)}{D}\right] - 1$$
$$A_i = \frac{V_i}{(t_i^{e_i})}$$

If $e_i < 1$, then we have a condition where the constraints may be satisfied by setting $e_i = 1$, since the distance integral (equation 3) will be less than D under this new assignment.

Another problem is that, especially in a walking gait, the knees joints remain rigid as the legs pass through the midstance position, causing the body to rise along an arc to a maximum (rather than a minimum) halfway through its support phase. The solution PODA adopts is to sinusoidally interpolate between this arc-like trajectory and trajectory resolved using the piece-wise speed curve as a function of the ratio of non-support over support intervals during each gait cycle [see fig. 9]. In this way, walking (having a zero ratio) has a maximum height at midstance and jumping has a minimum height at midstance with running somewhere in between.

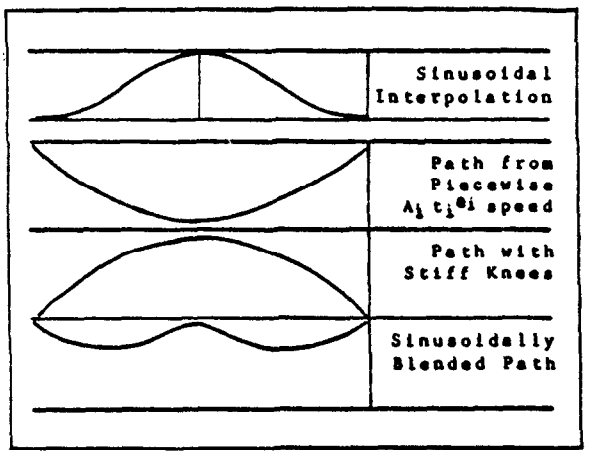

Figure 9:

Although the above interpolation scheme works reasonably well, we need a more physically motivated model. We need to know the dynamic control laws which are involved in the establishment of these vertical body trajectories during the locomotion of real animals.

4.2 Horizontal Body Motion

The desired horizontal path taken by the animal is interactively designed by the animator with a cubic spline (Catmull-Rom or B-spline). Given the desired body speed along different parts of the curve, PODA may calculate the desired positions and velocities along it using the numerical arc-length calculation discussed above.

The body's ability to turn and speed up is modeled in terms of the number of feet on the ground during the support intervals and the magnitude of the maximum achievable acceleration each supporting leg may apply to the body as a whole. We call this maximum achievable acceleration the leg's *impulse*. This parameter, which may be interactively set by the animator, indirectly determines the sense of weight of the animal in relation to the strength of its legs. The maximum achievable acceleration of the body at any given instant is the sum of the impulses of its supporting legs.

PODA determines the desired acceleration at a given frame through velocity error feedback, that is, by subtracting the desired velocity from the current velocity at that frame. The horizontal acceleration of the body is then computed as the minimum of its maximum achievable acceleration and the desired acceleration at each frame. Note that if, at any instant, there are no supporting legs, the maximum achievable acceleration will be zero, leaving a minimum of zero. Therefore, as we expect, the body will move at constant horizontal speed while in the air.

Given the acceleration, Euler integration is applied at each frame to solve for the new horizontal velocity and position.

4.3 Angular Motion: Turning

The modeling of angular motion during locomotion in PODA is currently restricted to rotations about the yaw and roll axis — that is, turning and banking respectively. The horizontal motion pass outputs the precise positions of the animal, so the desired yaw angle may by found by calculating the direction of movement from changes in horizontal position. Turning is achieved by solving Newton's equations of motion to bring the body to the measured yaw angle at the beginning of each body support interval. If we assume a constant acceleration during each support period, we have:

$$\ddot{\theta}_i = \frac{\theta_{i+1} - \theta_i - \dot{\theta}_i(t_s + t_{ns})}{(1/2)t_s^2 + (t_s)(t_{ns})}$$

$$\dot{\theta}_i = \ddot{\theta}_{i-1} t_s$$

where t_s = support interval
t_{ns} = following non-support interval
$i+1$ = start of next support interval
i = start of current support interval
$i-1$ = start of previous support interval

4.4 Angular Motion: Banking with dynamic stability

An animal is dynamically stable when the the projections of the gravitational and inertial forces sum to zero about the foot's point of contact with the ground [10]. To maintain dynamic stability, a running animal must adjust its foothold calculation and banking angle to counteract the centrifugal force as it turns.

Let us consider the situation of an animal moving at speed s along a circle of radius r. The centrifugal force, by Newton's equations, will be a vector perpendicular to the circle having magnitude ms^2/r. If we assume that a planted foot will not slide due to friction, we wish to solve for the banking angle in which there is no net angular torque. This condition is reached when the gravitational force downward, mg, and the centrifugal force outward sum to a force vector which points directly down the banking angle [see fig. 10].

Therefore, the banking angle is:

$$\phi = \arctan\left[\frac{s^2}{Rg}\right]$$

and the change in foothold will then be:

$$sideStep = (hipHeight)\tan\phi$$

We generalize this solution to arbitrary curved paths by noting the the curvature of the path $K = 1/R$. PODA fits a spline through the path of the animals body in the horizontal plane. At the frames in which legs come to their midstance position, the curvature of the path is measured and the dynamically stable banking angle is computed according to the given equation with $K = 1/R$.

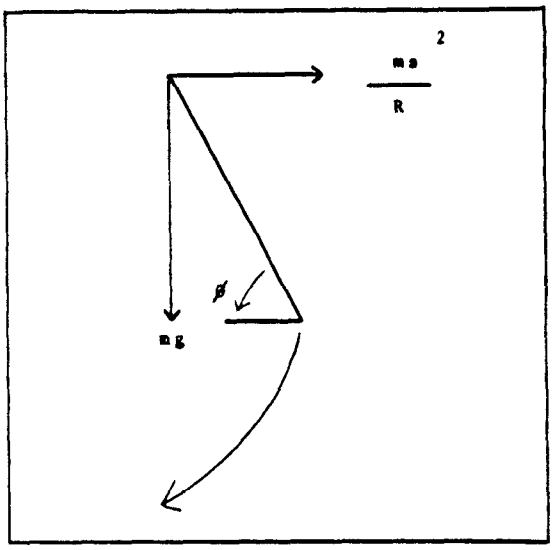

Figure 10:

The banking angle is then spline-interplated for frames which fall in between, using the midstance banking angles as control points.

5 Non-periodic Movement

In contrast to locomotion, expressive choreography, as well as most other coordinated activities, typically requires the capability for animating the body in a non-periodic fashion. We decided to focus on the problem of dance due to the challenge of creating motion which must be composed with a concern for the details or "micro-structure" of movement. The computer-animation of dance forces the issue of coordination and timing beyond building computational models for specific motor skills — we must identify primitives for the composition of arbitrarily complex forms of movement.

Our model for locomotion relies on as-

sumptions we can make about the patterns of movement: the feet move in the periodic rhythm of gaits, the arms (if the animal has arms) swing in opposition to the legs, the feet lift from the toe and land on the heel, the bending of the spine is synchronized to the motion of the pelvis. But expressive movement is much more open-ended in its possibilities. The body may move suddenly in any direction. During the support phases, the body height may vary dramatically — during the non-support phases, the entire body may assume several postures before returning to the ground [see fig. 11].

5.1 Limb and Body Trajectories

In order to define such free-form body movement, PODA supports the composition of *body trajectories*. A body trajectory, in effect, is nothing more than a sequence of body postures — it is the union of the spine, head, and various limb trajectories. The pelvis is also given a trajectory — when the feet are planted on the ground, the pelvis may be envisioned as an end-effector moving from its legs. Expressive movement may frequently involve both pelvic translations and rotations.

From the animator's standpoint, body trajectories are composed, remembered and recalled in the same fashion as limb trajectories: menu-driven interaction is provided for inserting, replacing, deleting, storing and retrieving postures and posture sequences.

At the time of this writing, we have implemented only the vertical body motion in the context of non-periodic motion. For continuous path motion, the horizontal and angular motion models we used for locomotion will apply, but the question of how to best simulate and control dramatic changes in horizontal and angular motion which may be synchronized with the placement of individual steps is still under consideration.

5.2 Vertical Body Motion from Body Trajectories

Each posture includes a specification of which legs are supporting the animal from the ground. As the animator "flips through" each posture in a sequence, the number of the posture and the support/non-support state of each leg are displayed in the viewing window (**s** means "support," and **m** denotes "moving, or non-supported") [see fig. 11]. Unsupported postures (having no supporting legs) will be performed while the animal is in the air.

Unlike locomotion, the height and overall form of the body at the instant of becoming airborne and at the instant of becoming earthbound may vary considerably ; e.g., hopping from a crouched position with the knees bent vs. hopping with the legs nearly fully extended. Therefore, the vertical heights of the supporting postures which immediately precede and follow the unsupported postures are used by PODA as a guide for simulating the vertical motion of the pelvis. We will call these *transition body postures*. For non-periodic motion, the support profile is constructed by recording the non-support periods which occur between these transition body postures.

The same constraints for vertical motion during locomotion hold here, but in this new context, the height of the body at liftoff and

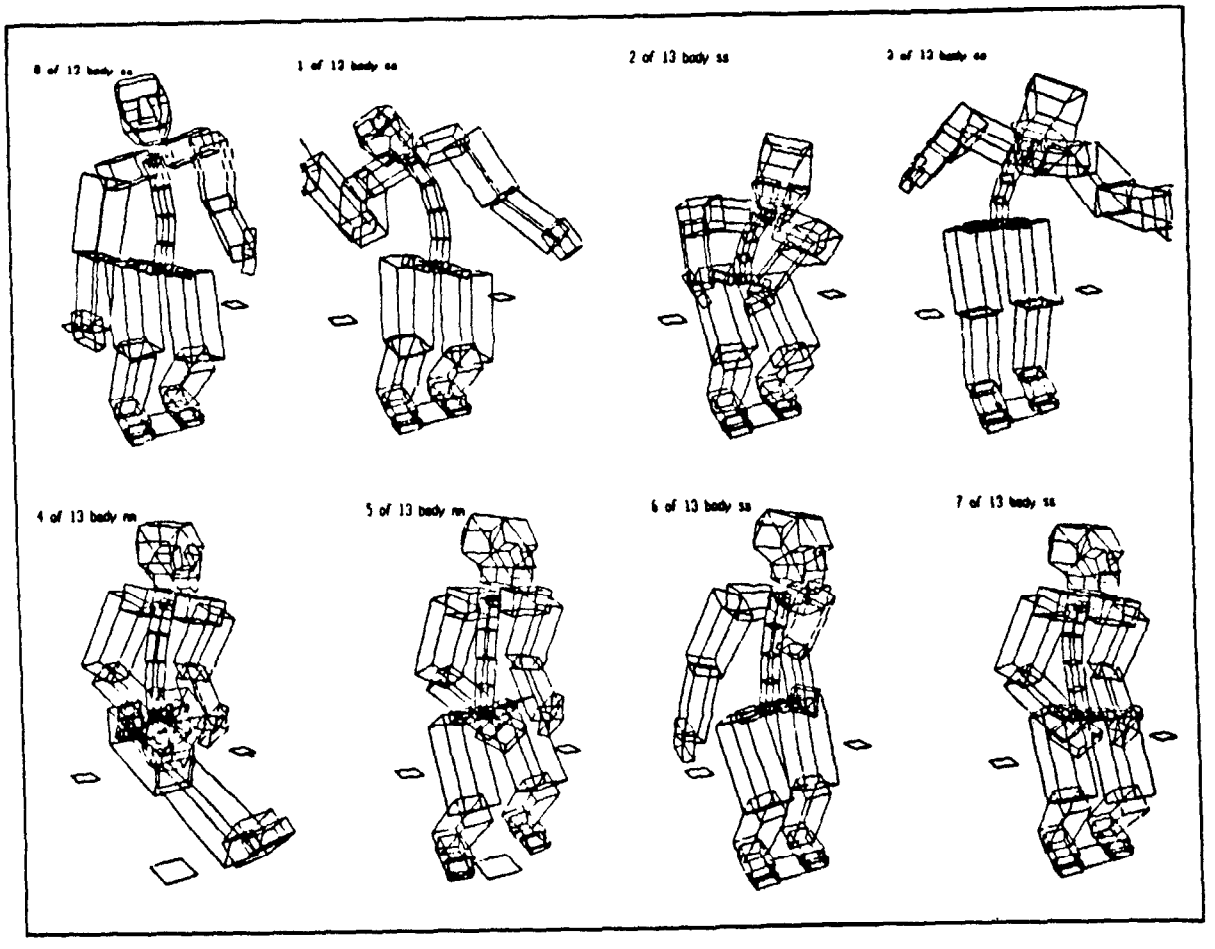

Figure 11: Sequence of eight body postures. Postures numbered 4 and 5 are in a non-supported state.

touchdown must match the heights at the transition postures which have been specified by the animator. The upward velocity at liftoff (and subsequent downward velocity as the body lands) may be solved for each pair of liftoff followed by touchdown transition body postures using:

$$V_{up} = g\left[(1/2)(t_{ns}) - \frac{h}{gt_{ns}}\right]$$

where g = gravitational acceleration
h = difference in height
t_{ns} = duration of non-support interval

All that remains is to find the vertical pelvis trajectories during the support phases. If the support phase is defined by only two body postures, both must be transition postures and PODA will use the given piece-wise speed function to find the support-phase pelvis trajectory. If there are more than two, the animator must have intended to modulate the height of the pelvis using the pelvis trajectory defined by the three or more body postures which take place during the support phase.

For this case, PODA sinusoidally interpolates between a trajectory of the form derived from the piece-wise speed function and the kinematically specified pelvis trajectory. The distance and interval variables of the piece-wise speed function are set to the differences in time and distance between transition postures and their neighboring support postures (the posture after the touchdown transition posture and the posture before the liftoff transition posture). In this way, the differences in height and time of these neighboring support postures may be used to vary the perceived muscular response of the animal for both counteracting its downward acceleration at the instant of impact and thrusting upward at the instant of becoming airborne.

6 Conclusion

The key point of this paper is that computational modeling of coordinated animal motion must simulate the anticipatory control strategies animals use in all facets of their movement. We discussed these strategies in the context of limb movement trajectory planning, and developed a means of defining these trajectories in a precise geometric fashion. Secondly, we described how our model for body and leg coordination makes use of optimization criteria in planning for the adaptive placement of feet and the shifting of gaits during locomotion. Finally, we demonstrated how an animal's body trajectory may be incrementally refined by a multiple pass planning process which gradually accumulates constraints on its three-dimensional degrees-of-freedom.

The second point is that an animator may design a framework of motion through the use of interactively manipulated posture sequences and interactively accessible dynamic and temporal constraints. These parameters provide a control mechanism for establishing the set of postural and rhythmical goals which the anticipatory control strategies embedded in the legged animal motion model are designed to meet.

7 Acknowledgements

The author wishes to thank the members and staff of the Computer Graphics Research Group, especially George Karl, Brian Guenter, Susan Amkraut, John Renner, Dave Haumann, Chris Wedge, Scott Dyer, Jeff ("Frog") Campbell, Charles Csuri, Tom Linehan, Kris Conrad, and my advisor, Bruce Weide, for their intellectual, emotional, and technical support. Special thanks to Juan Valdez.

References

[1] Whitney, D. E., "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine Systems,* **MMS-10**(2) pp. 47-53, June, 1969.

[2] Klein, C. A. and Huang, C. H., "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics,* **SMC-13**(2) pp. 245-250, March/April, 1983.

[3] Liegeois, A., "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Transactions on Systems, Man and Cybernetics,* **SMC-7**(12) December, 1977.

[4] Maciejewski, Anthony A., and Klein, C. A., "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments," *The International Journal of Robotics Research,* **4**(3) Fall 1985.

[5] Chang, Pyung H., "A Closed Form Solution for Inverse Kinematics of Robot Manipulator with Redundancy," *A.I. Memo 854,* March, 1986.

[6] Mortenson, Michael E., *Geometric Modeling,* John Wiley & Sons, Inc., 1985.

[7] Millman R., Parker G., *Elements of Differential Geometry,* Prentice-Hall Inc., 1977.

[8] Kochanek D., and Bartels, R., "Interpolating Splines with Local Tension, Continuity, and Bias Control," *SIGGRAPH proceedings,* 1984.

[9] Steketee S., Badler N., "Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control," *SIGGRAPH proceedings,* 1985.

[10] Vukobratovic, M. and Stokie, D., "Is Dynamic Control Needed in Robotic Systems, and, if So, to What Extent?," *The International Journal of Robotics Research,* **2**(2) Summer, 1983.

[11] Atkeson, C. G. and Hollerbach, J. M., "Kinematic Features of Unrestrained Arm Movements," *A.I. Memo 790,* July, 1984.

[12] Atkeson, C. G. and Hollerbach, J. M., "Characterization of Joint-Interpolated Arm Movements," *A.I. Memo 849,* June, 1985.

[13] Nelson, W. L., "Physical Principles for Economies of Skilled Movements," *Biological Cybernetics,* **46**(2) 1983.

[14] Morasso, Pietro, "Three Dimensional Arm Trajectories," *Biological Cybernetics,* **48**(3) 1983.

[15] Flash, Tamar, and Hollerbach, J. M., "Dynamic Interactions Between Limb Segments During Planar Arm Movement," *Biological Cybernetics,* **44**(1) 1982.

[16] Morasso, P. and Ivaldi, F. A. Mussa, "Trajectory Formation and Handwriting: A Computational Model," *Biological Cybernetics,* **45**(2) 1982.

[17] Wilhelms, Jane Patricia, "Graphical Simulation of the Motion of Articulated Bodies Such as Humans and Robots, with Particular Emphasis on the Use of Dynamic Analysis," Ph.d Dissertation, University of California, Berkeley, 1985.

[18] Armstrong, Wm., and Green, M. W., "The Dynamics of Articulated Rigid Bodies for Purposes of Animation," *The Visual Computer,* **1**(4) 1985.

[19] Armstrong, W. W., Green, M. and Lake, R., "Near-Real-Time Control of Human Figure Models," *Graphics Interface,* 1986.

[20] Brady, J. M., "Trajectory Planning," in: Brady, J.M., Johnson, T.L., Lozano-Perez, T., and Mason, M.T. (eds) *Robot Motion: Planning and Control.* Cambridge, MA. MIT Press, 1982.

[21] Girard, M., and Maciejewski, A., "Computational Modeling for the Computer Animation of Legged Figures," *SIGGRAPH proceedings,* 1985.